A Multilevel Cache Memory Architecture for Nanoelectronics

David Crawley
Image Processing Group, Dept. of Physics and Astronomy,
University College London, Gower Street, London WC1E 6BT United Kingdom.
d.crawley@ucl.ac.uk

Abstract

In this paper, we present a new multilevel cache memory architecture which uses only near-neighbour connections, thus eliminating long tracks and rendering the system suitable for nanoelectronic implementation. Operation of the memory is such that the most-recently accessed data is kept closest to the read-write port.

1. Introduction

In a conventional cache memory system [1] there are a small number (usually between 1 and 3) of cache levels positioned between the processor and main memory. The L1 cache is the fastest and smallest with L2 and L3 being commensurately larger and slower. When an attempt to read an object in memory is made, the object is first sought in the L1 cache, then in L2, L3 and finally in main memory. Cached data is stored in blocks consisting of several bytes. Cache relies on the *locality* of memory references in order to increase memory system performance - if access has been made to a given memory location then there is a high probability both it and nearby locations will be accessed in the near future.

Unfortunately, this form of memory system design may not be well-suited to a nanoelectronic implementation because of the need for long interconnection paths within each of the cache levels and within the main memory itself. As the width of a metal track on a die decreases, its resistance increases. This reduces the speed at which signals propagate, since both the capacitance of the load and the distributed capacitance of the track itself must be driven. Alternative nanoelectronic 'wires'- for example, those formed by lines of Quantum Cellular Automata (QCA) [2] are also unlikely to be suitable for the implementation of long communication paths because of similar delay problems, although the limitation in signal propagation speed is due to the need to operate the cells in an adiabatic switching regime, rather than from resistive-capacitive (RC) delays.

The primary objective of the scheme presented in this paper is to avoid the use of long interconnection paths.

2. Multilevel Cache Memory

The novel memory system we describe here may be considered as an *n*-level (L1-L*n*) series of caches. Unlike conventional cache memory schemes, all the levels are the same size and there is no 'main memory' - the multiple cache levels constitute the entire store.

Data is stored in blocks: each block holds a number of contiguous words together with the upper bits of their addresses.

When a request to access a location is made via the read/write port, the address of the block containing the desired word is passed up to each level where an associative store is searched for a matching block. The level holding the requested block passes it down to the read/write port and at the same time it is placed in level 1. The least-recently used block from level 1 is passed to level 2, the least-recently used block from level 2 is passed to level 3 and so forth, until the level which held the requested block receives the oldest block from the preceding level.

In this way, the most-recently used blocks are held in the level closest to the read/write port and the least-recently accessed blocks are held in the levels most distant from the read/write port.

2.1 Details of Operation

Figure 1 illustrates the logical structure of the memory system. Access to the memory is made via the Port Controller shown at the bottom of the structure. The Port Controller is connected to the storage elements via two paths: the 'Up' path shown on the right of the diagram and the 'Down' path shown on the left. Information in the memory is stored in structures which will be referred to as Memory Blocks or 'Blocks'. Each Block consists of the fields shown in Figure 2:

0-7695-0104-4/99 $10.00 © 1999 IEEE

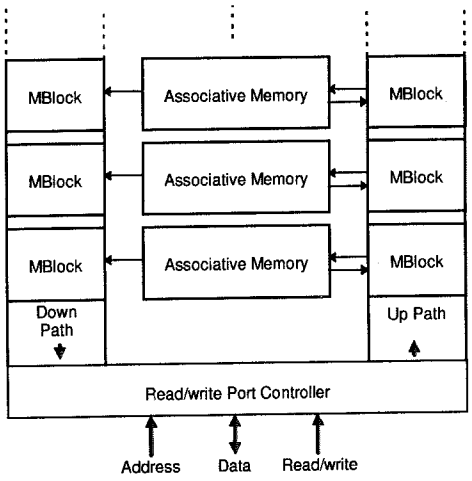

Figure 1 Memory Logical Structure

- A unique block address consisting of the upper address bits of each of the words in the data field.
- A data field which may contain a number of sequential words of information.
- Word address / age field. For templates, this field holds the lower bits address of the word being accessed - it indicates which word in the data field is to be written or read. For blocks stored in memory, this field indicates when the block was last accessed.
- A type field indicating the nature of the block. This may take any of the values shown in the figure, in particular this field is used to indicate whether the Block actually contains data or is a search template. The values of this field are described in more detail below.

When an a memory access is made by asserting the appropriate signals at the read/write port, the following sequence of events occurs:

1. The port controller constructs a 'template' block. The given address is split into two components - the upper (more significant) bits are used to fill the block address field, while the lower (less significant) bits are stored in the word address / age field. If data is being written then the value given on the data lines is used to set the corresponding word in the data field. The type field is set to indicate a read or write template, as appropriate.
2. The port controller passes the template block to the first level via the 'Up' path.
3. At the first level, a copy of the template block is passed to the second level via the 'Up' path. At the same time, an attempt is made to match the block address of the given template with those stored in the associative memory. Subsequent levels in the structure repeat this process.
4. When a match occurs, the type field of the template is examined. If the template specifies a read operation, then the located block is copied into the 'Down' path with its type set to 'located' and the Age/Word address field copied from the template block. The block in the associative memory has its type field set to 'replaceable'. If the template specifies a write operation, a similar sequence of events occurs, except that the word specified in the template block is used to overwrite the corresponding word in the data field of the block passed on the 'Down' path.
5. The located block is passed down from level to level via the 'Down' path. The presence of the 'located' type field causes each level to end its search for the located block (if the search has not already completed) and instead begin a search for the oldest block in the associative store by examination of the Age fields of each of the stored blocks.
6. When the oldest block in a level is found, it is copied via the 'Up' path to the level above with the type field set to 'demoted'. The corresponding block in the associative store is marked as 'replaceable'. When a 'demoted' block is received at a level via the 'Up' path, its contents are copied into the block marked as 'replaceable' in the associative store. The Age fields on all blocks in the associative store are then updated so that the new block has an age of zero.

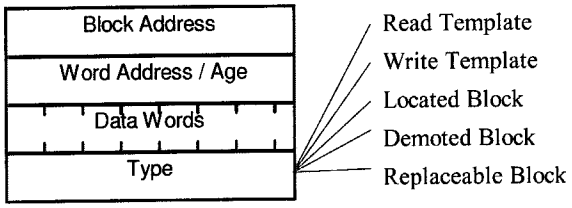

Figure 2 Memory Block Structure

Conclusions

We have described a novel multilevel cache memory architecture suitable for nanoelectronic implementation. The architecture eliminates the need for long wires within the memory structure, whilst the cached nature of the design attempts to minimize data access time. The associative memory could be implemented using shift registers constructed using the QCA[2] devices. Timing analysis and simulation of this structure have been performed and will be described in a subsequent paper. Future developments include examination of a binary tree version of the memory architecture.

[1] Hennessy, J. L. and Patterson, D. A., *Computer Architecture A Quantitative Approach,* Morgan Kaufmann Publishers Inc., CA, 1990.

[2] Lent, C.S., and Tougaw, P.D., "A Device Architecture for Computing with Quantum Dots," *Proc. IEEE.* vol.85, no.4, April 1997, p.541-57.